from the editor

Editor in Chief: Hakan Erdogmus ■ National Research Council Canada
hakan.erdogmus@computer.org

25 Years of Software

Hakan Erdogmus

2008 marks the 25th anniversary of IEEE Software. This final issue of 2008 comes to you with many special celebratory features compiled by several of Software’s editors and staff. We hope to take you back in time to accentuate notable milestones and accomplishments, viewed from different perspectives, in our industry and in the history of software development.

Looking Back
Twenty-five years ago, I was a sophomore, studying computer engineering in a freshly established department in a developing country. My university had limited computing resources. Progress meant upgrading from typing programs at a card puncher to a few hours of weekly allowance in front a text-based mainframe terminal.

I remember being ecstatic about having switched that year from using cumbersome punch cards to glittering, green Univac terminals to complete our programming assignments. (Yes, punch cards. I can say this only in hindsight, but I now consider myself fortunate to have used them. And not only for nostalgic novelty, but also because punch cards forced us to think hard about the correctness and composition of our programs before we could compile them). Of course, we had little idea about how far behind the times we were and how little we knew of such things as software design, requirements, and process. Perhaps we’d learned a few things about data structures and modularity in a Fortran course the year before, and that was that. How pathetic was it, for example, that I hadn’t heard of Smalltalk! Still, every bit of software technology we encountered was new to us, and we considered ourselves pretty lucky to be leading the next generation of computing professionals in our home country (pity, many of us didn’t stick around!). This was a time when several of Software’s veteran editors and contributors were doing serious work unimaginable to me at the time. So I find myself humbled in their presence.

Things of course are different now for higher-education institutions throughout the world. Software development innovations and knowledge are instantly available to the masses, easily accessible to both educators and students wherever they are, with virtually no adoption barriers. Knowledge in every imaginable niche and all sorts of tools and technology (a great many of them free) are ours to pick and choose from. The problem is rather one of separating the perpetually worthy from the lightning fad, of deciding what and how much to know, and of choosing an optimal solution from a vast number of alternatives. Time and our capacity to absorb new information are the only remaining excuses for lagging behind. Granted, they’re good ones. And all of us perpetually do lag behind. So, ironically,
Software Board Changes

We welcome Laurence Tratt to the editorial board. Laurence replaces Sophia Drossopoulou as the new associate editor in chief for programming languages and paradigms. Sophia wished to retire early to look after other priorities. During her tenure, Sophia worked hard to grow our programming-language coverage, sharing her wisdom and perspective and reaching out to the academic community to attract accessible contributions with practical implications. Many thanks to Sophia for her efforts and advice.

Laurence, a senior lecturer at Bournemouth University, has been a dynamic member of Software’s advisory board since 2005. An educator, researcher, consultant, developer, and language designer, Laurie is most notably the chief creator of the Converge programming language and an expert in domain-specific languages and software modeling. In 2007, he was a guest editor of the magazine’s very successful focus section on dynamically typed languages. Laurence is a proactive member of the programming languages and software development communities, with many contributions to open source projects and international standards.

The Way We Think about Design

Yes, we’ve made progress. For one, the notion of software design has taken a firm hold since the assembly- and Fortran-dominated days of software development. We have new vocabulary, concepts, power tools, and technologies at our disposal that enable us to think about and shape our software. Although the basic elements of program construction haven’t changed much, the way we talk about our constructions has definitely evolved. Rebecca Wirfs-Brock reflects upon that transformation in her special anniversary column.

The Human-Facing Side of Software

Even after I had immigrated to Canada to pursue grad studies, I still knew nothing about the human-facing side of software design. Back then the more complicated the software I used and the harder it was to use, the smarter and more victorious I felt. This was at a time when the likes of Larry Constantine and Alan Cooper were getting pre-occupied by what makes software useful, what makes users tick with pleasure and twitch with disgust. Jeff Patton’s interview with pioneer Alan Cooper takes us back to the origins of the thinking behind today’s modern interaction design practice.

Evolution of Requirements Practices

As for dealing with software requirements, that field has come a long way from its roots in structured analysis and specification techniques. In his column, Neil Maiden talks about how those early issues gradually led the way to new emphases...
centered on requirements elicitation, tool support, and human factors. The outcome was to turn the tackling of software requirements into a central activity as well as a sought-after specialty that demands a distinct set of skills.

**Software Measurement in the Mainstream?**
Not exactly, at least not yet. But as Voice of Evidence guest contributor Shari Lawrence Pfleeger explains, software measurement has also been walking the slow but steady path, from being a science of software metrics to an integral part of software development. She identifies four areas of future focus for software measurement: dealing with uncertainty, human and organizational aspects, dealing with changes in the environment, and more widespread incorporation of heuristic approaches.

**A Roster of Software Technologies**
Yes, the past 25 years have seen a crowded parade of all sorts of software technologies. Some of them have faded into oblivion, but others, or at least certain of their key elements, have stuck around. Christof Ebert’s column reviews this history. Christof uses the term “technology” broadly to also include important concepts and paradigms that left their mark on our industry. His interpretation, augmented by input from the magazine’s boards, clumps technologies into groups and graphs their evolution from inception to maturation.

**The Good Old Days of Programming**
Sure, we do we have plenty of power tools at our disposal nowadays, but that doesn’t mean we’ve abandoned the plain hammers and screwdrivers of the trade, many from the “nix toolbox, to rust on dusty shelves. On the contrary, these tools, and the time-tested technique of stitching them together with pipes and filters, still make up many seasoned programmers’ main arsenal. In his Tools of the Trade piece, guest columnist Brian Kernighan talks about the persistence and value of the old ways in the era of the super integrated development environment and the refactoring browser.

**Reflections, Reflections …**
That’s not all, though. Our contributing writer Greg Goth solicited comments from several luminaries and Software’s past editors in chief. The luminaries were asked to comment on major accomplishments of the past 25 years. The only constraint they were given was that they couldn’t name one of their own accomplishments. The editors in chief were asked to reflect on their then-goals for the magazine and the specific challenges that they faced during their four-year tenure. Each also commented on the period’s events and trends that for them signified important advances.

Greg also compiled a rudimentary time line as a visual accompaniment to the special anniversary pieces. We later tweaked it with extra input from the boards. The time line comprises selected milestones from the history of software development and some events that we think represent the larger computing context in which
those milestones were achieved. Treat it as food for thought. Let us know if you spot any grand omissions or undeserved inclusions—and don’t forget to explain why.

Recurring Themes
If you read all the pieces, you may start to tire of hearing about the impact of the softer issues, those that relate to human and organizational aspects of software development. This thread pops up over and over again. Well, it’s because there’s much to say on the topic. Listen to the experts: we’ve only begun to understand the role of soft factors in attacking serious technical challenges.

Domain specificity is another recurring theme. And yes, that’s also likely to keep coming back. The limitations of one-size approaches are hitting us over the head, again and again. As our experts point out, the need to exploit the particular structure and idiosyncrasies of a complex problem’s context and domain is becoming urgent.

More to Come
We are planning to continue our celebration of Software’s fruitful existence in the first issue of the new year with more special pieces. In particular, look for Dan O’Leary’s analysis of the all-time top-cited Software articles.

Happy silver anniversary, Software! ☛